N87-16743

D1-61
15 P
[illegible]

Implementation of a Data Management Software System for SSME Test History Data

Kenneth Abernethy
Associate Professor of Computer Science
Furman University
Greenville, South Carolina

ABSTRACT

This report documents the implementation of a software system for managing SSME test/flight historical data. The implementation was completed on a VAX 11/780, but will be transferred to an IBM 3084QX in the near future. The software system uses the database management system RIM7 for primary data storage and routine data management, but includes several FORTRAN programs, described here, which provide customized access to the RIM7 database. The consolidation, modification, and transfer of data from the database THIST, implemented in 1984 using DATATRIEVE, to the RIM7 database THISRM is discussed.

The RIM7 utility modules for generating some standard reports from THISRM and performing some routine updating and maintenance are briefly described. In addition, two errors in the RIM7 software are documented and procedures for avoiding their encounter are detailed. In particular, errors are identified in the RIM7 routine for reloading a database to recover file space marked for deletion by various RIM7 functions and the RIM7 routine for building B-tree indices for attributes. These errors were encountered in v7.0 update 12 of RIM7 running on a VAX 11/780 under the operating system VMS 4.4, and it is unknown if they will be present when the IBM 3084QX version of RIM7 is employed.

The FORTRAN accessing programs described include programs for initial loading of large data sets into the database, capturing data from files for database inclusion, and producing specialized statistical reports which cannot be provided by the RIM7 report generator utility.

An expert system tutorial, constructed using the expert system shell product INSIGHT2, is described. Finally, a potential expert system, which would analyze data in the database, is outlined. This system could use INSIGHT2 as well and would take advantage of RIM7's compatibility with the microcomputer database system RBase 5000.

INTRODUCTION

The software system described in this report was implemented following a detailed feasibility and requirements study, which was completed in 1984. The design of the system followed the design and use of a prototype system, which was implemented in 1984-5 and whose use has provided several important design modifications for the present system.

Major functional requirements which came out of the feasibility study included:

a. Ad hoc queries must be supported.
b. Database structures must be easily modified.
c. Report generation must be flexible and non-complicated.
d. Interface to existing data files should exist.
e. Graphical output capability is highly desirable.

Several database management systems (DBMS) in existence at MSFC were explored during the summer of 1984 and evaluated relative to criteria which included the above requirements. One of these systems used a hierarchical data model, another used a network data model, and two used a relational data model. Based on the relatively high importance of criteria (a) and (b) above, it was decided that a relational data model would be best suited as the core data management tool for the proposed system. It was further decided that a prototype system using an existing MSFC database management system should be designed.

A SSME test history database, THIST, was designed and implemented using Digital Equipment's DATATRIEVE file management system. During late 1984 and early 1985, data for the prototype was collected from various sources and loaded into THIST. In addition to providing a vehicle for the initial test history data collection, this prototype provided user orientation and familiarization with the data management and query capabilities of a relational DBMS. It further provided a check on the data model chosen for representing the SSME test history data.

Originally it was planned to implement the full scale software system in the summer of 1985. However, at that time MSFC was in the process of obtaining a new center wide technical computing system, EADS (Engineering Analysis and Data System). It was decided that it would be highly desirable to implement the test history data management system on EADS for several reasons. This implementation would provide wide accessibility to the system and would permit a more natural interface with SSME test data which will be collected and stored on EADS hardware. Perhaps more importantly, the EADS processors, considerably more powerful than the VAX and UNIVAC processors currently in use at MSFC, would be available for relational query processing. The flexibility and user-friendliness of a relational database system extracts a price in terms of efficiency, and the use of the DATATRIEVE prototype had exposed inefficient relational query processing as a probable serious problem as the test history database increases in size. Because of these considerations, the implementation of the full SSME test history database was postponed until the summer of 1986, when EADS hardware would be in place and its software support capabilities would be better known.

1. IMPLEMENTATION OF THE CORE DATABASE.

The DATATRIEVE database THIST was constructed as a prototype only. Since DATATRIEVE is a file management system as opposed to a true relational DBMS, it was suspected that it would not handle relational algebra commands efficiently enough for certain ad hoc queries, especially as the data files grew. Experience with the prototype confirmed this. Additionally, it was known that DATATRIEVE could not be implemented on EADS. As a consequence, the first step in implementing the full database environment on EADS was to select the core DBMS.

Several DBMS's were considered. One of the relational systems evaluated during the requirements phase of the project was ORACLE. Indeed, ORACLE is an excellent system and would be an ideal choice except that the EADS version of ORACLE costs over $70,000. The DBMS chosen for inclusion in the original EADS software was System 2000. Although System 2000 is not a relational model system (it uses the hierarchical model), it does have a 4th generation relational type query language. Also among its advantages was the fact that it is already resident on EADS and hence would require no software purchase. However, the System 2000 procedure for modification of the database structure is rather complex, and since one of the primary functional requirements is that the database be easily restructured, System 2000 was rejected. The third DBMS considered, and the one chosen for the application, was RIM5 (and its successor RIM7). RIM5 was developed in the late 1970's under NASA contract by Boeing Computer Services, and is available through COSMIC, NASA's software distribution center. It has a number of features which make it particularly attractive for the SSME test history application. For example, it has a graphical output capability, includes scientific data types, has a fully relational structure, and includes a powerful query language and a good report generator utility. It was decided to evaluate RIM5 more thoroughly and so the VAX version was acquired for this purpose, after it was discovered that the IBM version of RIM5 would not run under the IBM operating system used by the EADS processors. Boeing Computer Services has developed a derivative product, RIM7, which will run on EADS and which is reasonably priced at $15,000. Thus the plan was to evaluate RIM5 (which was obtained free) to decide whether or not RIM7 should be purchased for EADS.

Using the RIM5 FORTRAN Application Program Interface, programs were written to read the THIST data from the DATATRIEVE created VAX RMS files and load it into a RIM5 database, THISRM, whose relations match those of THIST. Experience with the THIST data model had suggested that three of these relations could be profitably combined into a single relation. This consolidation would allow the production of several often needed reports without having to use time-expensive relational algebra commands to combine the relations at the time of report generation. Before doing this database restructuring, however, it was necessary to align the three relations so that they contained data for exactly the same tests, as was not the case with the corresponding DATATRIEVE relations which had been created and loaded at different times. To accomplish this, the RIM5 relational algebra commands were used to generate 7 data files which contained the test numbers (key attributes) that were missing from one or more of the relations in THISRM. Then more FORTRAN programs were written to use those files of missing test numbers to "pad" the appropriate relations, filling the non test number attributes with the special RIM5 symbol -0- which indicates that data is missing. When producing summary reports involving missing data so marked, RIM5 then excludes this data from counts, averages, etc. Finally, using RIM5's relational algebra again, relations were combined to restructure the database THISRM. A total of 1314 engine tests/flights were represented in the database as of July 1, 1986, and the stored data occupied approximately 1.5 megabytes of disk space.

2. EVALUATION OF RIM.

The transfer of the prototype database THIST to the RIM5 database THISRM was relatively straightforward, and the Application Program Interface proved to be a useful and well designed module. However, when attempts were begun to restructure THISRM, as described above, several problems with the RIM5 software did arise. Errors were generated when certain relational algebra commands necessary for the restructure were attempted. The sources of these errors were not apparent immediately. After discovering that a VAX installation of RIM7 was available at MSFC, it was decided to port the entire application to RIM7 to investigate whether the difficulties encountered were inherent in RIM5. Unfortunately, the same difficulties arose in the use of RIM7.

Following several weeks of diagnostic testing, the sources of the errors were isolated to some extent, and work-around strategies were devised. One source of error proved to be the RELOAD module in RIM. When rows are deleted from relations, or when entire relations are removed from the database, the deleted space is not actually recovered for reuse, but is flagged, or marked, as deleted. Of course, this can lead to very poor file and disk space utilization. Unfortunately, the problem is quite unavoidable because it is often necessary to remove relations from the database. For example, if we have a relation that we wish to sort (as would often be the case after new data has been appended), the way to accomplish this in RIM is to create a sorted copy of the relation, remove the relation itself, then rename the sorted relation to the old relation name. This is an easy procedure to employ, but it leaves unrecovered disk space equal to the size of the sorted relation. The recovery of space marked for deletion is done by executing the RIM command RELOAD. However, after RELOAD is executed following a sort operation on a relation of moderate size (say 1000 rows of 200 bytes each), several of the RIM commands no longer work properly. In particular the PROJECT command will replace the parent relation by the projected relation in the database schema, thereby losing access to the parent relation data. The JOIN command has a similar malfunction in it. Curiously, these erroneous results do not appear to occur when the same actions are taken on small relations and databases.

The problems described above can be avoided (apparently) by using the UNLOAD command in place of the RELOAD command. The UNLOAD command creates a separate copy of the entire database (schema and indices included) in a file named COMPFIL.DAT, recovering record space marked for deletion in the process. After executing the UNLOAD command, the database files themselves must be deleted (using the VAX VMS delete command), and then the entire database must be reconstructed using the INPUT command and the file COMPFIL.DAT. While this process is a bit more cumbersome and takes about twice the time of a RELOAD command execution, it does seem to recover unused space cleanly without adverse effects on any other RIM commands or the database contents.

A second error source is the BUILD KEY command. Again, with a moderately large database, after executing a BUILD KEY command, which creates a B-tree index for a chosen attribute and relation, subsequent commands can produce erroneous results. For example in attempting a JOIN command using the indexed attribute, part of the relation involved appears to be missing. That is, the JOIN simply ignores, or can't find, a large part of the relation it is seeking to join to another. To avoid this situation, one needs to exit RIM with an EXIT command immediately after the BUILD KEY command executes, thus forcing the newly created index to be copied from working space to the database index file. If other commands intervene between the BUILD KEY and EXIT commands, errors are likely.

Both of the problems described above have been reported to Boeing Computer Services and work is underway there to correct them. In the meantime, the fixes given above seem to avoid the errors. However, it would not be surprising to see other problems arise, especially in connection with the use of the BUILD KEY command, and until the actual errors in the code itself have been identified and corrected by Boeing, caution should be exercised in the use of BUILD KEY, and RELOAD should be altogether avoided.

The RIM7 report generator appears to be very flexible and more than adequate for most of the anticipated application report generation. Report definitions become part of the database, and reports can be printed at any time. Each report is associated with a unique relation whose definition must be in the database schema at the time the report is defined. The data contents of this relation, however, may be modified by the PROJECT, SELECT, or other commands prior to printing a report. Report definitions can be modified using a built-in report editor, although this editor is rather cumbersome to use. Other features of the report generator provide considerable flexibility. For example, control breaks can be inserted at various points within a report, including user specified attribute value changes. At these breaks, totals and subtotals of an attribute or expression, including count, sum, average, minimum, and maximum, can be displayed. On the other hand, reports, such as statistical analysis reports, requiring more detailed computational manipulation of data, cannot be defined using this utility and must be generated using the Application Program Interface, as discussed in the next section.

The update utility allows user-tailored input screens to be defined for data input. A certain amount of on-screen documentation can be provided as a guide to the screen user. There is no computational capability, and so if any preprocessing of data (such as unit conversion) is needed before storing the data, this would have to be provided via the Application Program Interface.

Additional RIM7 features include plotting capability, backup/logging and recover facilities, and downward file compatibility with the microcomputer database system RBase 5000. Simple X-Y line plots of up to ten Y attributes plotted against one X attribute are available. Plots are supported on Tektronics 4010 and 4014 terminals, and on Hewlett-Packard 2623 and 2647 terminals. Database files can be downloaded for RBase 5000 use via an option in the UNLOAD command.

In summary, although there are some problems with the current version of RIM7, it has a great many features which make it particularly appropriate for the SSME test history application. It is assumed that Boeing will soon correct these documented problems, but even if those corrections are are not immediate, the work-arounds suggested seem to be adequate for now. Considering especially the cost factor, it was decided to proceed with development of the SSME data management system using RIM7 as the core DBMS.

3. THE FORTAN ACCESSING PROGRAMS.

In order to provide for increased usefulness and data accessibility for THISRM, several FORTRAN programs were constructed making use of the RIM7 Application Program Interface. These programs provide functions that are not available with RIM7 proper. A brief description of the programs follow, and an example program is developed to illustrate the way the Application Program Interface is used. No code listings are included, but these are available from the author or his NASA counterpart.

Program THLOAD is a program for reading and appending (or initially loading) large data sets to THISRM. If data is loaded for only some of the THISRM relations, the program automatically places the relevant test numbers (key values) in all the other relations (provided these key values aren't already present). Attribute values in these other relations are filled with RIM7's special "missing data" symbol. The program allows data to be added to such padded relations. This procedure guarantees that the relations will remain the same size and that missing data will be easy to spot within the database. The program is menu-driven and provides the user with complete instructions for its use. It is documented in-line so that it can be easily modified to conform to any restructuring of the database schema.

Program THCAPT is a straightforward modification of THLOAD to allow for capturing data from files to be loaded into THISRM. The program is designed so that the only changes necessary to THCAPT to allow capture from different data files is a modification of certain READ and FORMAT statements. The program is interactive and provides the user with the appropriate instruction for its use/modification. It provides for global or attribute particular data capture.

Program THSTAT provides for certain standard statistical summaries of database attributes which cannot be generated through the RIM7 report utility. Again, the program is interactive and menu-driven, so that the user can request one or more of several categories of statistical analysis, on as many or as few attributes as desired. User instruction on using the SELECT command within RIM7 to choose a file for analysis is provided if the user desires.

The following example develops a brief FORTRAN program which will illustrate the techniques needed to write programs utilizing the RIM7 Application Program Interface. The example assumes that the appropriate database relation exists already. The program could be written in any language that supports calls to external FORTRAN subroutines. Data transfer from a file to the database is accomplished by reading the file data into program variables, and then using those variables as parameters in calls to the RIM7 subroutines for placing the data into the database. If the language used supports a record data type (as do COBOL and Pascal), the data can be read into a record variable, whose structure matches that of the RIM7 relation row, and then transferred by that variable name via a call to the appropriate RIM7 subroutine. In FORTRAN, we must simulate a record structure by using an array and EQUIVALENCE statements. We illustrate with a simple example.

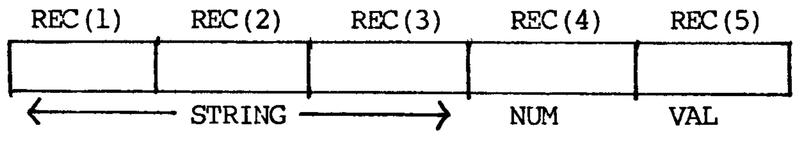
Suppose our file has the following format:

text 10 characters	integer	real with format dd.ddd with decimal implied in storage

We would then make the following declarations:

```
      DIMENSION REC(5)
      CHARACTER*10 STRING
      INTEGER NUM
      REAL VAL
      EQUIVALENCE (REC(1),STRING), (REC(4),NUM), (REC(5),VAL)
```

This sets up in memory the following variable overlay:



Note: 4 bytes (characters) per array location, hence we need 3 locations for STRING.

We now (assuming the file has been opened as unit 9), read the data as follows:

```
      READ(9,100) STRING,NUM,VAL
  100 FORMAT (A10,I,F5.3)
```

and store this information into the database by:

```
      CALL RMLOAD(1,REC).
```

Additional RIM7 subroutines needed are RMOPEN (to open the appropriate database), RMUSER (to specify the database password - if it has one), RMFIND (to position pointer to the desired relation), and RMCLOS (to close the database when operations are completed). Database access error codes (whose meanings are given in the RIM7 User's Guide) are communicated to our program via the COMMON BLOCK variable RMSTAT. This variable should be checked after each database access since it is the only way errors (except for fatal system errors) are communicated to the program.

All of these ideas are illustrated in the following complete program which would transfer the file HARDW.DAT to the relation HARDWARE in the RIM7 database THISRM which has password SSME.

```
C  PROGRAM  TRANSFERS FILE HARDW.DAT TO RELATION  HARDWARE  IN  RIM7
C  DATABASE THISRM.
C
C
C  DATA WILL BE READ INTO VARIABLE NAMES AND THEN ASSOCIATED WITH AN
C  ARRAY BY EQUIVALENCE STATEMENTS,  SINCE THE RIM7 DATABASE ACCESS
C  ROUTINE RMLOAD REQUIRES AN ARRAY AS PARAMETER.  THE ARRAY DBREC
C  WILL CONTAIN THE DATABASE RECORD.
C
C
C
C
C  **************************    MAIN    ***************************
C
C
C  THE INTEGER RMSTAT IS A RIM7 DATABASE COMMUNICATION BUFFER
C  VARIABLE, FOR RECEIVING STATUS CODES FROM THE RIM7 DATABASE
C  MANAGEMENT SYSTEM AFTER EACH DATABASE ACCESS.  COMMUNICATION IS
C  ESTABLISHED VIA THE DATABASE RESERVED COMMON BLOCK RIMCOM.
C
C
      DIMENSION DBNAME(2), DBPASS(2)
      COMMON/RIMCOM/RMSTAT
      INTEGER RMSTAT
```

```
C  THE ARRAY DBNAME HOLDS THE NAME OF THE RIM7 DATABASE TO BE
C  LOADED AND THE ARRAY DBPASS HOLDS THE DATABASE PASSWORD.  THESE
C  NAMES MUST BE ASSIGNED IN HOLLERITH FORMAT.  BOTH COMPONENTS OF
C  THE ARRAYS MUST BE ASSIGNED -- EVEN IF THE NAME IS LESS THAN 5
C  CHARACTERS LONG, THE SECOND COMPONENT MUST BE ASSIGNED A BLANK.
C
      DBNAME(1)=4HTHIS
      DBNAME(2)=2HRM
      DBPASS(1)=4HSSME
      DBPASS(2)=1H
C
C  RMOPEN IS THE RIM7 ROUTINE TO OPEN THE DATABASE.  RMUSER IS THE
C  ROUTINE TO GIVE THE PASSWORD.  A NON-ZERO VALUE FOR RMSTAT
C  INDICATES THAT AN ERROR HAS OCCURRED IN USING A RIM7 DATABASE
C  ACCESS SUBROUTINE.
C
      CALL RMOPEN(DBNAME)
      IF (RMSTAT .NE. 0) GOTO 999
      CALL RMUSER(DBPASS)
      IF (RMSTAT .NE. 0) GOTO 999
C
C
C  MOVDAT IS THE PROGRAM SUBROUTINE WRITTEN TO TRANSFER DATA TO
C  THE DATABASE.
C
C  RMSTAT IS WRITTEN WHEN AN ERROR CONDITION IS DETECTED.
C
C  RMCLOS IS THE RIM7 ROUTINE TO CLOSE THE DATABASE.
C
      CALL MOVDAT
      GOTO 1000
  999 WRITE (6,9001) RMSTAT
 9001 FORMAT ('ERROR HAS OCCURRED.  RMSTAT IS ', I5)
 1000 CALL RMCLOS
      END
C
C
C  **************** SUBROUTINE MOVDAT ***************************
C
C  THIS SUBROUTINE MOVES DATA FROM THE FILE HARDW.DAT TO THE THISRM
C  RELATION HARDWARE.
C
      DIMENSION DBREC(14),RNAME(2)
      COMMON/RIMCOM/RMSTAT
      INTEGER RMSTAT
```

```
C  THE FOLLOWING VARIABLE NAMES AND DECLARATIONS CORRESPOND TO
C  THE ATTRIBUTE NAMES AND DECLARATIONS IN THE THISRM RELATION
C  HARDWARE.
C
      CHARACTER*11 TESTNO
      INTEGER      POS
      CHARACTER*6  PARTD
      CHARACTER*7  SN
      CHARACTER*8  UN
      CHARACTER*13 PARTNO
C  THE FOLLOWING EQUIVALENCE STATEMENTS DEFINE THE DATABASE RECORD
C  WITHIN THE ARRAY DBREC BY ASSOCIATING THE VARIABLES INTO WHICH
C  THE FILE DATA IS READ, WITH THE ARRAY ELEMENTS.  EACH ARRAY
C  ELEMENT CAN BE ASSOCIATED WITH A SINGLE INTEGER OR REAL, OR
C  WITH 4 CHARACTERS OF TEXT.
C
      EQUIVALENCE (DBREC(1),TESTNO), (DBREC(4),POS),
     %(DBREC(5),PARTD), (DBREC(7),SN), (DBREC(9),UN),
     %(DBREC(11),PARTNO)
C
C  THE VAX RMS FILE TO BE TRANSFERRED IS HARDW.DAT.  THE
C  FOLLOWING STATEMENT OPENS THAT FILE FOR READING.
C
      OPEN (UNIT=9, FILE='HARDW.DAT', READONLY, STATUS='OLD')
C
C  RMFIND IS THE RIM7 ROUTINE USED TO LOCATE THE DATABASE
C  RELATION INTO WHICH WE WILL LOAD THE FILE HARDW.DAT.
C  NOTE THAT THE RELATION NAME MUST BE DEFINED IN HOLLERITH
C  FORMAT IN ORDER FOR THE PARAMETER TYPES TO MATCH IN THE
C  CALL TO RMFIND.
C
      RNAME(1)=4HHARD
      RNAME(2)=4HWARE
      CALL RMFIND(1,RNAME)
      IF (RMSTAT .NE. 0) GOTO 999
C
C --------------   LOOP TO TRANSFER DATA ------------------
C
C  RMLOAD IS THE RIM7 ROUTINE TO LOAD A RECORD (ARRAY) INTO THE
C  CURRENT (AS DEFINED BY THE ASSOCIATED RMFIND CALL) RELATION.
C  THE PARAMETER 1 IDENTIFIES A POINTER  AND SHOULD BE THE SAME AS
C  THE NUMERICAL PARAMETER USED IN THE ASSOCIATED RMFIND CALL.
C
C  NOTE THAT THE LOOP IS EXITED VIA THE READ STATEMENT --
C  -- 'END=101' FORCES A JUMP TO STATEMENT 101 WHEN AN
C  END-OF-FILE CONDITION IS ENCOUNTERED.
```

```
      K=0
  200 READ (9,100,END=101) TESTNO,POS,PARTD,SN,UN,PARTNO
      CALL RMLOAD(1,DBREC)
      IF (RMSTAT .NE. 0) GOTO 999
      K=K+1
  100 FORMAT (A11,I1,A6,A7,A8,A13)
      GOTO 200
C
C  ERROR REPORTING IS DONE BY WRITING RMSTAT
C
  999 WRITE (6,9001) RMSTAT
 9001 FORMAT ('ERROR HAS OCCURRED IN MOVDAT.  RMSTAT IS ', I5)
      GOTO 1000
  101 WRITE (6,9002) K
 9002 FORMAT (I5, 'RECORDS TRANSFERRED TO RELATION HARDWARE.')
 1000 CLOSE (UNIT=9)
      RETURN
      END
```

4. USE OF EXPERT SYSTEMS WITH THISRM.

An important potential application for THISRM involves the use of an expert system to analyze database contents. The expert system shell product INSIGHT2, which runs under MS/DOS on an IBM PC/XT or compatible, provides the capability for creating rule-based expert systems without the necessity of user produced code. INSIGHT2 has a built-in interface capability with DBASE II data files. In order to use INSIGHT2 for building a THISRM data analysis expert system, one could proceed as follows. Using RIM7's UNLOAD command with the MICRO option, RIM7 database contents can be transferred to an IBM PC in RBase 5000 format. A Pascal program (Turbo Pascal with special DBASE II database access procedures added is included in INSIGHT2) could be written to transform the RBase 5000 files to the DBASE II compatible files necessary for INSIGHT2 access. Then the desired expert analysis system could be constructed using INSIGHT2. Of course, the files to be transferred from RIM7 to the expert system could first be tailored by use of RIM7's relational algebra commands. In this way the query language in RIM7 can be combined with the rule-based analysis of INSIGHT2 to provide exceptionally powerful data query capability. Because of time constraints, such an application could not be completed this summer.

INSIGHT2 has been used to construct a rule-based tutorial for the THISRM database environment. Instruction is provided on the basic capabilities of RIM7 as well as on the use of the auxiliary FORTRAN programs. The user proceeds through the tutorial by answering questions which provide the tutorial with information to navigate through its rule set. In this way, a tutorial session is tailored to the individual user's current needs and responses. The tutorial was constructed not only to provide user documentation for THISRM in a highly accessible format, but also to provide an immediate example of rule-based expert system development using INSIGHT2.

REFERENCES

1. Abernethy, K., "Using a Database Management System for Modelling SSME Test History Data," NASA/ASEE 1984 Summer Faculty Research Reports, Marshall Space Flight Center.

2. Boeing RIM Application Program Interface Guide, The Boeing Co., 1985, Seattle, Washington.

3. Boeing RIM Forms Data Entry Guide, The Boeing Co., 1985, Seattle, Washington.

4. Boeing RIM Report Writer User's Guide, The Boeing Co., 1985, Seattle, Washington.

5. Boeing RIM User's Guide, The Boeing Co., 1985, Seattle, Washington.

6. INSIGHT2 Reference Manual, Level Five Research, Inc., 1985, Melbourne, Florida.